NRL Memorandum Report 6381

Fortran Codes for Computing the Space-Time Correlations of Turbulent Flow in a Channel

R. A. HANDLER AND F. ROSENTHAL

Laboratory for Computational Physics and Fluid Dynamics

December 30, 1988

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 6381	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b OFFICE SYMBOL (If applicable) Code 4420	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Fortran Codes for Computing the Space-Time Correlations of Turbulent Flow in a Channel

12 PERSONAL AUTHOR(S)
Handler, R.A. and Rosenthal, F.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Interim	FROM 6/88 TO present	1988 December 30	48

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Turbulence, Correlations, Symmetries, Fortran

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Computer codes have been developed which compute the space-time correlations in turbulent channel flow. The two-point spatial correlation function is computed by Fourier transforming the appropriate wavenumber spectrum. The wavenumber spectrum is computed by averaging over an appropriate number of ensembles and using geometric symmetries to improve the smoothness of the results. Space-time correlations are computed in a similar manner.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Robert A. Handler	22b TELEPHONE (Include Area Code) (202) 767-2457 22c OFFICE SYMBOL Code 4420

DD Form 1473, JUN 86 Previous editions are obsolete SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603

CONTENTS

FORTRAN PROGRAMS FOR COMPUTING THE SPACE-TIME CORRELATIONS OF TURBULENT FLOW IN A CHANNEL

1. Introduction

Recently, several groups have performed direct numerical simulations of turbulent flow in a channel at low Reynolds numbers. These have included the high resolution (fully resolved) calculation of Kim, Moin, and Moser (1987) and the low resoluton calculations of Gilbert and Kleiser (1987). These simulations generate enormous quantities of data which have been used to investigate the validity of current theories of turbulence, to study mechanisms for the production of turbulence in wall-bounded flows, and to develop new turbulence models. However, before such detailed investigations can be undertaken, it is important to first check some basic statistics of the turbulence such as the mean velocity profile, turbulence intensities, and Reynolds stress profiles. Beyond these one-point statistics, it is also of considerable value to have a rapid means of computing the two-point statistics so that the length and time scales of the turbulence can be determined. In this report, we describe FORTRAN codes which we have developed which enable rapid evaluation of the two point correlations from a given dataset. Datasets were created in two forms. In the first, we have taken 'spatial snapshots' (realizations) of the flow. That is, at a given time in the calculation, we have stored the three components of the velocity at each collocation point designated by its coordinates x,y, and z, which are respectively the streamwise, spanwise, and wall-normal coordinates. In general, we have saved the realizations at time intervals on the order of or larger than the largest correlation time of the flow so that we may say that the dataset consists of a number of statistically independent snapshots. In the second type

Manuscript approved October 6, 1988.

of data, we have stored, at certain fixed locations above the wall, the three components of velocity in the horizontal planes at equally spaced time intervals. The time between saves for the planar data is generally significantly less than the correlation time of the flow. We will call such datasets 'planar' datasets. In Section 2 we describe codes which produce the two-point correlation function of turbulence from a 'snapshot' dataset and in Section 3 we describe the code which produces the space-time correlation function from a 'planar' dataset.

2. Computation of the Complete Two-Point Correlation Function for Turbulent Channel Flow from Spatial Realizations

A. Mathematical Preliminaries and Definitions

Before describing the computer codes in detail, we briefly discuss certain mathematical definitions and the methods used to compute the two-point correlation function. (In the following discussions, we use the variables x_1, x_3, and x_2 interchangeably with the notation x, y, z, and bold face notation indicates a vector quantity.) For any given flow field, $\mathbf{u}(\mathbf{x}, t)$, we obtain a set of statistically independent realizations by sampling the field at times separated by at least the characteristic correlation time of the flow. We define the Fourier coefficients of any such realization by:

$$\mathbf{\Phi}(n, m, x_2) = \sum_{l=0}^{N_1-1} \sum_{k=0}^{N_3-1} \mathbf{u}(l, k, x_2) e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi mki}{N_3}},$$

where

$$i = \sqrt{-1},$$

$$n = 0, ... \frac{N_1}{2} + 1, \tag{1}$$

and

$$m = \frac{-N_3}{2}, ... \frac{N_3}{2} - 1.$$

The expression above defines collocation points $(x_1)_l$ and $(x_3)_k$, and wavenumbers $(k_1)_n$ and $(k_3)_m$ as follows:

$$
\begin{aligned}
(x_1)_l &= l \times L_1/N_1, \\
(x_3)_k &= k \times L_3/N_3, \\
(k_1)_n &= n \times \frac{2\pi}{L_1}, \\
(k_3)_m &= m \times \frac{2\pi}{L_3},
\end{aligned}
\tag{2}
$$

in which L_1 and L_3 are the domain lengths in the streamwise and spanwise directions respectively. In equation 1 time serves merely as a parameter which may be used to identify any given flow realization and is therefore suppressed. We now follow Sirovich (1987), who suggests that the geometric symmetries of a flow should be exploited to increase the effective number of realizations that are used in the averaging process. For the case of channel flow, the equations of motion are invariant with respect to wall normal reflection, spanwise reflection, and 180 degree rotation about the streamwise axis. Applying these symmetries to the spectral representation $\mathbf{\Phi}^0$, where we use the superscript 0 to refer to the representation before reflections and rotations have been applied, yields three additional representations of the flow which are given by:

$$
\begin{aligned}
\mathbf{\Phi}^1 &= \{\Phi_1^0(n,m,-x_2), -\Phi_2^0(n,m,-x_2), \Phi_3^0(n,m,-x_2)\} \\
\mathbf{\Phi}^2 &= \{\Phi_1^0(n,-m,x_2), \Phi_2^0(n,-m,x_2), -\Phi_3^0(n,-m,x_2)\}, \\
\mathbf{\Phi}^3 &= \{\Phi_1^0(n,-m,-x_2), -\Phi_2^0(n,-m,-x_2), -\Phi_3^0(n,-m,-x_2)\}.
\end{aligned}
\tag{3}
$$

In these expressions, the superscripts 1, 2, and 3 refer to vertical reflection, spanwise reflection, and streamwise rotation symmetries respectively. The subscripts continue to represent the coordinate directions as previously defined. The (complex) spectrum $\Psi_{\alpha\beta}$ can now be computed from:

$$\Psi_{\alpha\beta}(n,m,x_2,x_2') = \langle \frac{1}{4}\sum_{p=0}^{3} \overline{\Phi_\alpha^p}(n,m,x_2)\Phi_\beta^p(n,m,x_2')\rangle, \tag{4}$$

where

$$\alpha,\beta = 1,2,3.$$

The overbar designates complex conjugate, and the brackets represent an average over all the available realizations. We note that in addition to the increase in the effective size of the data base by a factor of four, the symmetries have the additional advantage of reducing the size of $\Psi_{\alpha\beta}$ also by a factor of four. A factor of two comes from the spanwise symmetry which insures that $\Psi_{\alpha\beta}$ is unique for $n > 0$ and $m > 0$. A second factor of two comes from the vertical reflection symmetry which gives unique values of the spectrum for $-1 < x_2 < 0$ and $-1 < x_2' < 1$. The two point correlation function , $R_{\alpha\beta}$, can now be obtained from:

$$R_{\alpha\beta}(l,k,x_2,x_2') = K \sum_{n=0}^{\frac{N_1}{2}-1} \sum_{m=-\frac{N_3}{2}}^{\frac{N_3}{2}-1} \Psi_{\alpha\beta}(n,m,x_2,x_2') e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi mki}{N_3}}, \tag{5}$$

where

$$K = \frac{1}{\sqrt{R_{\alpha\alpha}(0,0,x_2)R_{\beta\beta}(0,0,x_2')}}.$$

We note that care must be taken in the calculation of $R_{\alpha\beta}$ since $\Psi_{\alpha\beta}$ is computed only for $n, m \geq 0$. That is, the inherent symmetries in $\Psi_{\alpha\beta}$ must be taken into account when equation 5 is used. In particular, the following symmetries must hold:

$$\begin{aligned}
\Psi_{11}(n,-m,x_2,x_2') &= \Psi_{11}(n,m,x_2,x_2'),\\
\Psi_{22}(n,-m,x_2,x_2') &= \Psi_{22}(n,m,x_2,x_2'),\\
\Psi_{33}(n,-m,x_2,x_2') &= \Psi_{33}(n,m,x_2,x_2'),\\
\Psi_{12}(n,-m,x_2,x_2') &= \Psi_{12}(n,m,x_2,x_2'),\\
\Psi_{13}(n,-m,x_2,x_2') &= -\Psi_{13}(n,m,x_2,x_2'),\\
\Psi_{23}(n,-m,x_2,x_2') &= -\Psi_{23}(n,m,x_2,x_2').
\end{aligned} \tag{6}$$

A detailed check on the computation of $\Psi_{\alpha\beta}$ was made by computing the energies (mean square values) of each velocity component by summing $\Psi_{\alpha\beta}$ over m and n. The comparison of these values with the known energies was exact and will be described in a future report.

B. Description of the codes KLEXP and KLANA

The computation of the correlation function $R_{\alpha\beta}$ defined by eqation 5 is computed by a two-stage process. First, the spectral function $\Psi_{\alpha\beta}$ defined by equation 4 is computed and saved. This computation is performed using the code KLEXP. Then the spectral function is used to calculate $R_{\alpha\beta}$ using the code KLANA. Originally, the interest was primarily to perform an eigenfunction decomposition (Lumley(1970)) of turbulent flow in a channel. To perform such a decomposition, the code KLEXP was written to determine $\Psi_{\alpha\beta}$ which was of greatest interest for our purposes. Later it was decided that the the real space function $R_{\alpha\beta}$ would also be of interest so that the code

KLANA was written for the purpose of computing it. All codes discussed in this report are listed in the Appendix.

B.1 Description of the code KLEXP

The code KLEXP listed in the Appendix was written to compute $\Psi_{\alpha\beta}$ from a data set which consists of 30 realizations of channel flow turbulence. In this particular case, there were 7 data files containing 4 realizations and one containing 2. On each file, the three components of vorticity were also stored but were skipped each time the data was accessed. After each realization of the velocity field is read into memory, certain baseline statistics such as the mean and variance of each velocity component are computed. These values, which are calculated from the raw (real space) data, can later be compared to the values obtained from the spectral function $\Psi_{\alpha\beta}$ as a check on the calculation. Next, the horizontal transform of the three components of velocity is performed using the subroutine YXFOUR. This calculation is represented mathematically by equation 1. The six components of $\Psi_{\alpha\beta}$ are then calculated from the raw spectral representation, $\mathbf{\Phi}_\alpha$, using the subroutine MATRIX. In this routine, for each spectral realization of the flow, the product $\overline{\Phi}_\alpha \Phi_\beta$ is computed and averaging is performed over the symmetry groups given in equation 3. We note that the parameter 'is' is used to distinguish the four possible ways (which will be described below) in which the symmetries need to be applied.

We now describe in detail how the six components of $\Psi_{\alpha\beta}$ are computed. In computing Ψ_{11} we first compute r^0_{11} and r^1_{11} which are given by:

$$r^0_{11}(n, m, x_2, x_2') = \overline{\Phi_1}(n, m, x_2)\Phi_1(n, m, x_2'), \tag{7}$$

and

$$r^1_{11}(n, m, x_2, x_2') = \overline{\Phi_1}(n, m, -x_2)\Phi_1(n, m, -x_2').$$

We then apply the spanwise reflection symmetry to r_{11}^0 and r_{11}^1 using the subroutine SYM1. This yields r_{11}^2 and r_{11}^3 which are given by:

$$r_{11}^2(n,m,x_2,x_2') = r_{11}^0(n,m,x_2,x_2') + r_{11}^0(n,-m,x_2,x_2'),$$

and

$$r_{11}^3(n,m,x_2,x_2') = r_{11}^1(n,m,x_2,x_2') + r_{11}^1(n,-m,x_2,x_2'). \tag{8}$$

Finally, the function Ψ_{11} is computed from:

$$\Psi_{11}(n,m,x_2,x_2') = r_{11}^2(n,m,x_2,x_2') + r_{11}^3(n,m,x_2,x_2'). \tag{9}$$

This procedure incorporates all three group symmetries given in equation 3. The same procedure is used to compute Ψ_{22} and Ψ_{33}.

To obtain the other three components of the correlation matrix, minor modifications of the procedure outlined above are needed. In the computation of Ψ_{12} we must take into account the change in sign of the wall-normal velocity. This is expreesed by :

$$\Psi_{12}(n,m,x_2,x_2') = r_{12}^2(n,m,x_2,x_2') - r_{12}^3(n,m,x_2,x_2'). \tag{10}$$

In the computation of Ψ_{13} we must modify the relations in equation 8 since there is a sign change in the spanwise velocity component when the spanwise reflection symmetry is used. The resulting expressions for r_{13}^2 and r_{13}^2 are given by:

$$r_{13}^2(n,m,x_2,x_2') = r_{13}^0(n,m,x_2,x_2') - r_{13}^0(n,-m,x_2,x_2'),$$

and

$$r_{13}^3(n,m,x_2,x_2') = r_{13}^1(n,m,x_2,x_2') - r_{13}^1(n,-m,x_2,x_2'). \tag{11}$$

In calculating Ψ_{23} we must take into account both the sign changes in the wall-normal velocity component and the spanwise velocity component. This results in the following relations:

$$r^2_{23}(n,m,x_2,x_2{}') = r^0_{23}(n,m,x_2,x_2{}') - r^0_{23}(n,-m,x_2,x_2{}'),$$

$$r^3_{23}(n,m,x_2,x_2{}') = r^1_{23}(n,m,x_2,x_2{}') - r^1_{23}(n,-m,x_2,x_2{}'), \tag{12}$$

and

$$\Psi_{23}(n,m,x_2,x_2{}') = r^2_{23}(n,m,x_2,x_2{}') - r^3_{23}(n,m,x_2,x_2{}'). \tag{13}$$

The final result , $\Psi_{\alpha\beta}$, is then scaled appropriately and diagnostics are applied to check the answer. A good check is to determine whether the energy in $\Psi_{\alpha\beta}$, obtained by integrating over all wavenumbers, is equal to the energies computed from the raw (real space) fields. The final result is then saved on disc. No unnecessary information has been computed or stored in this calculation and, as noted previously, the application of the symmetries reduces the storage requirements by a factor of four.

B.2 Description of the code KLANA

Code KLANA uses the spectral results $\Psi_{\alpha\beta}$ produced by KLEXP as described in equation 4 to compute the correlations $R_{\alpha\beta}$ described in equation 5. Before the main calculation, all six spectral functions are read from disc. The main loop (do 8000) computes the correlation functions one at a time and writes each result to disc successively. Since the calculation of each of the six correlations is identical except for the application of the symmetries described in equation 6, a description of the calculation of R_{11} will serve as a description for the other five. For each pair of indices 'iz' and 'izp' which designate wall-normal distances, a two dimensional array 'phi' is extracted from the four dimensional spectral function Φ_{11}. Since the resultant correlation must

be real, the known symmetry properties of Φ_{11} are applied to the array 'phi' in subroutine SETPHI to generate the array 'phisym'. The two dimensional Fourier transform is then performed in subroutine RIJ to complete the calculation. Note that the parameter 'ij' in the call to RIJ determines which of the symmetries given in equation 6 will be applied. The final result is placed into the larger array 'rrij', which is then normalized using the proper K defined in equation 5. These correlation functions and the normalization factors which have been used are written to a disk file for later plotting.

3. Calculation of the Space-Time Correlation Function from Planar Data

A. Mathematical Definitions

Before describing the details of the FORTRAN code that has been written to compute the space-time correlation function, we first describe the method of calculation. Given the planar realization $\mathbf{u}(x_1, x_3, t)$, we define the following spectral representations:

$$\gamma_\alpha(n,m,t) = \sum_{l=0}^{N_1-1}\sum_{k=0}^{N_3-1} u_\alpha(l,k,t) e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi mki}{N_3}},$$

$$\theta_\alpha(n,j,x_3) = \sum_{l=0}^{N_1-1}\sum_{k=0}^{N_t-1} u_\alpha(l,k,x_3) e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi kji}{N_t}}, \tag{14}$$

$$\pi_\alpha(m,j,x_1) = \sum_{l=0}^{N_3-1}\sum_{k=0}^{N_t-1} u_\alpha(l,k,x_1) e^{\frac{2\pi mli}{N_3}} e^{\frac{2\pi kji}{N_t}},$$

and

$$\alpha = 1, 2, 3.$$

We note that x_2 is suppressed in equation 14 since it is assumed to be fixed in these calculations. We also define the (discretized) time and frequency as follows:

$$t_n = \frac{T}{N_t} n,$$

$$\omega_n = \frac{2\pi}{T} n \tag{15}$$

and

$$n = 0, 1, ..., N_t - 1.$$

T is the total time in any given realization. In these calculations we apply the spanwise symmetry to γ and π in the following way:

$$\gamma^1 = \{\gamma_1^0(n, -m, t), \gamma_2^0(n, -m, t), -\gamma_3^0(n, -m, t)\},$$

and

$$\pi^1 = \{\pi_1^0(-m, j, t), \pi_2^0(-m, j, t), -\pi_3^0(-m, j, t)\}. \tag{16}$$

where the subscripts and superscripts have been defined in Section 2. We can then compute the (complex) spectra defined by:

$$\Gamma_{\alpha\beta}(n, m) = \langle \frac{1}{N_t} \sum_{k=0}^{N_t-1} \frac{1}{2} \sum_{p=0}^{1} \overline{\gamma_\alpha^p}(n, m, t_k) \gamma_\beta^p(n, m, t_k) \rangle,$$

$$\Theta_{\alpha\beta}(n, j) = \langle \frac{1}{N_3} \sum_{k=0}^{N_3-1} \overline{\theta_\alpha^p}(n, m, (x_3)_k) \theta_\beta^p(n, m, (x_3)_k) \rangle, \tag{17}$$

$$\Pi_{\alpha\beta}(m, j) = \langle \frac{1}{N_1} \sum_{k=0}^{N_1-1} \frac{1}{2} \sum_{p=0}^{1} \overline{\pi_\alpha^p}(m, j, (x_1)_k) \pi_\beta^p(m, j, (x_1)_k) \rangle,$$

and

$$\alpha, \beta = 1, 2, 3.$$

We note again that no symmetry has been applied in the calculation of $\Theta_{\alpha\beta}$ as indicated in equation 17. In summmary, we compute the appropriate spectral functions by first averaging over the appropriate symmetries, then averaging over a coordinate , and finally averaging over the available realizations.

We then define, as before, the two point correlation functions by:

$$R^a{}_{\alpha\beta}(l,k) = K_a \sum_{n=0}^{\frac{N_1}{2}-1} \sum_{m=-\frac{N_3}{2}}^{\frac{N_3}{2}-1} \Gamma_{\alpha\beta}(n,m) e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi mki}{N_3}},$$

$$R^b{}_{\alpha\beta}(l,k) = K_b \sum_{n=0}^{\frac{N_1}{2}-1} \sum_{j=0}^{N_t-1} \Theta_{\alpha\beta}(n,j) e^{\frac{2\pi nli}{N_1}} e^{\frac{2\pi jki}{N_t}}, \tag{18}$$

$$R^c{}_{\alpha\beta}(l,k) = K_c \sum_{m=-\frac{N_3}{2}}^{\frac{N_3}{2}-1} \sum_{j=0}^{N_t-1} \Pi_{\alpha\beta}(m,j) e^{\frac{2\pi mli}{N_3}} e^{\frac{2\pi jki}{N_t}},$$

and

$$K_a = \frac{1}{\sqrt{R^a{}_{\alpha\alpha}(0,0) R^a{}_{\beta\beta}(0,0)}},$$

$$K_b = \frac{1}{\sqrt{R^b{}_{\alpha\alpha}(0,0) R^b{}_{\beta\beta}(0,0)}},$$

$$K_c = \frac{1}{\sqrt{R^c{}_{\alpha\alpha}(0,0) R^c{}_{\beta\beta}(0,0)}}. \tag{19}$$

B. Description of code WAVSPEC

The Fortran code WAVSPEC permits the analysis of data of the 'planar' type as described in the Introduction. This data consists of the three components of velocity u_1, u_2, and u_3 , the streamwise, spanwise, and normal components of the vector $\mathbf{u}(x_1, x_3, t)$. The code listed in the Appendix was written to compute correlations from a dataset consisting of seven files in which horizontal planes of the three components of velocity and the vorticity are stored. Each file contains 512 time steps. After each realization (i.e. a file consisting of 512 time steps) is read from disc using the subroutine READ, the computations performed in equations 14-17 are performed by subroutine HFENS.

(Array 'tmp' in subroutine READ is a dummy used to skip over the values of vorticity contained in the dataset.)

The input to subroutine HFENS is the raw planar data, and the output consists of the updated (ensemble averaged) spectra given by Γ, Θ, and Π defined in equaton 17. We describe only the calculation of Γ in HFENS, since the calculation of Θ, and Π is virtually the same. The first loop in HFENS (do 100) computes the average over time as indicated in equation 17. The second loop (do 70) extracts the two dimensional arrays (the y-x array) from the three dimensional data (y-x-t arrays). Each two dimensional array is then brought into wavenumber $(k_3 - k_1)$ space using the subroutine HFFT1. The subroutine ENSAV1 then computes the square of the magnitude of the Fourier coefficients and applies the spanwise reflection symmetry. This completes the calculation of Γ. As indicated in equation 17, no symmetry is applied in the the calculation of Θ.

As a check on the calculation, we observe that if we sum Γ, Θ, and Π over wavenumber and frequency, the results must be identical. This sum is the mean square value (the energy) of the velocity component. This serves as a check on the calculations and therefore, following the action of HFENS, the total energy for each of the three spectra are computed using the routines ENERG1, ENERG2, and ENERG3. The results of these calculations, which will be given in a subsequent report, show that the energies were equal to within machine roundoff error. After the energies have been computed, they are used to normalize the spectral results.

The spectra Γ, Θ, and Π are then used to compute the space-time or space-space correlations given in equation 18. This is accomplished by first calling SETPH1, SETPH2, and SETPH3, each of which arranges the spectral data in an appropriate form for subsequent Fourier transformation. The Fourier transforms are performed

by the routines RIJ1, RIJ2, and RIJ3, the correlations are normalized as indicated in equation 18, and the results are saved on disc.

Conclusions

We have described computer codes that have been written to compute the relevant correlation functions and their corresponding spectral representations from stochastic velocity data which has been obtained from direct simulations of channel flow turbulence. The codes we have written can be used to analyze data that is in one of two forms : (1) spatial 'snapshots' of the flow , and (2) 'planar' datasets. In the first case we compute the relevant correlations in a two step process. First, we compute the four-dimensional spectrum given in equation 4 in which we average over all available realizations and also incorporate the relevant symmetries. This intermediate result is of particular importance since it forms the basis for a Karhunen-Loeve or eigenfunction expansion to which we referred in Section 2. This calculation is performed by the code KLEXP. These results are then used to compute the (real-space) correlation function given in equation 5 using the code KLANA. For the second case, using data of the 'planar' form, the relevant spectra given in equation 17 and correlations given in equation 18 are computed by implementing the code WAVSPEC.

As we mentioned in the Introduction, along with the theoretical interest that exists in computing these multidimentional statistics, these statistics also serve as diagnostic tools which can be used to check the very large data bases that are generated from direct numerical simulations. A description and some preliminary interpretations of the results of these calculations will be given in a subsequent report.

Aknowledgements

This work was supported by the Fluid Dynamics Task Area of the Naval Research Laboratory. The authors would like to aknowledge the assistance of Dr. Ken Ball and Professor Lawrence Sirovich of Brown University in clarifying the use of symmetries in these calculations.

References

1. Kim,J., Moin,P., and Moser,R., 1987 Turbulence statistics in fully developed channel flow at low Reynolds number.*J. Fluid Mech.***177**, 133-166.
2. Gilbert,N., and Kleiser,L., 1987 Low-resolution simulations of transitional and turbulent channel flow. In Turbulent Shear Flows 6 (ed. F.Durst et.al.), Springer.
3. Sirovich,L., 1987b Turbulence and the dynamics of coherent structures. *Q. Appl. Math.* **45**, 573-582.
4. Lumley,J.L., 1970 Stochastic Tools in Turbulence. Academic Press.

Appendix

Here we list the codes KLEXP, KLANA, and WAVSPEC described in the text. It is hoped that the text description together with the annotations within each code will enable a user to make modifications for his particular application. Note that the fft routines CFFT2, RCFFT2, and CRFFT2 are library subroutines which are generally available on the Cray-XMP.

```
      program klexp
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   ----   THE PURPOSE OF THIS CODE IS TO COMPUTE THE                      C
C  CORRELATION MATRIX FOR A 3-D  TURBULENT CHANNEL FLOW.                   C
C  THE RESULTS OF THE CALCULATION ARE USED AS INPUT TO AN                  C
C  EIGENVALUE SOLVER FROM WHICH WE CAN DETERMINE THE KARHUNEN-             C
C  LOEVE EXPANSION FUNCTIONS. THE CORRELATION MATRIX IS OF THE             C
C  FORM UiUj(m,n,z,zprime), WHERE THE INDICES I AND J REPRESENT THE        C
C  VELOCITY COMPONENETS (1=STREAMWISE 2=WALL NORMAL 3=SPANWISE),           C
C  M IS THE SPANWISE WAVENUMBER, N, THE STREAMWISE, AND Z IS THE WALL      C
C  NORMAL COORDINATE.NOTE THAT THERE CAN ONLY BE SIX INDEPENDENT           C
C  MEMBERS OF UiUj, AND ALSO, BECAUSE OF SYMMETRY CONSIDERATIONS           C
C  WE NEED ONLY COMPUTE 1/2 OF EACH OF THESE MEMBERS. ALSO NOTE THAT       C
C  WE ONLY NEED TO COMPUTE FOR POSITIVE M AND  N .                         C
C      IN THE CHANNEL DATA USED HERE THERE WERE 16, 64 AND 33 GRID         C
C  POINTS IN THE 1, 2 AND 3 DIRECTIONS RESPECTIVELY.IN SPECTRAL SPACE      C
C  THERE ARE 7 NON-ZERO STREAMWISE WAVENUMBERS AND 23 SPANWISE             C
C  MODES. THE TRUNCATION IS DUE TO THE USE OF DEALIASING IN THE            C
C  ORIGINAL CHANNEL FLOW CALCULATION.                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      parameter(nz=32,ny=64,nx=16)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
      parameter(nyd2=ny/2,nyd21=ny/2+1)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
      parameter(nfiles=8)
      common/param/fac,fac1
      dimension u1(nz1,ny,nxp2),u2(nz1,ny,nxp2),u3(nz1,ny,nxp2)
      dimension om1(nz1,1,1)
      dimension u1u1(nyd2,nxp2,nzd21,nz1),u2u2(nyd2,nxp2,nzd21,nz1)
      dimension u3u3(nyd2,nxp2,nzd21,nz1),u1u2(nyd2,nxp2,nzd21,nz1)
      dimension u1u3(nyd2,nxp2,nzd21,nz1),u2u3(nyd2,nxp2,nzd21,nz1)
      dimension u1m(nz1),u2m(nz1),u3m(nz1)
      dimension u1rms(nz1),u2rms(nz1),u3rms(nz1)
      dimension u12rms(nz1),u13rms(nz1),u23rms(nz1)
      dimension s1(nzd21),s2(nzd21),s3(nzd21)
      dimension s12(nzd21),s13(nzd21),s23(nzd21)
C  SET CERTAIN USEFUL SCALING PARAMETERS
                call setup
        nsym = 4
        nens = nsym*30
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  FOR INPUT WE HAVE 7 FILES EACH OF WHICH CONTAINS 4 INDEPENDENT
C  REALIZATIONS OF THE FLOW AND AN 8TH FILE THAT HAS TWO REALIZATIONS.
C  THE FIRST 28 REALIZATIONS ARE SPACED 100 VISCOUS TIME UNITS
C  APART AND THE LAST TWO, ARE  50 VISCOUS TIME UNITS APART.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      do 1 nrec = 1,nfiles
            lu = 10+(nrec-1)
         if (nrec .le. 7)  then
               ntimes = 4
                 else
               ntimes = 2
         endif
      do 1 nread = 1,ntimes
           read(lu)(((u1(k,j,i),k=1,nz1),j=1,ny),i=1,nx)
           read(lu)(((u3(k,j,i),k=1,nz1),j=1,ny),i=1,nx)
           read(lu)(((u2(k,j,i),k=1,nz1),j=1,ny),i=1,nx)
           read(lu)(((om1(k,j,i),k=1,nz1),j=1,1),i=1,1)
           read(lu)(((om1(k,j,i),k=1,nz1),j=1,1),i=1,1)
```

```
                read(lu)(((om1(k,j,i),k=1,nz1),j=1,1),i=1,1)
CCCCCCCCCCCC   COMPUTE MEAN VALUES      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC  THIS IS BEING DONE MERELY FOR DIAGNOSTIC PURPOSES. THAT IS,
CC    WE ARE COMPUTING THE MEAN AND RMS FOR EACH OF THE THREE FIELDS.
                call mean(u1,u1m)
                call mean(u2,u2m)
                call mean(u3,u3m)
                call zms(u1,u1,u1rms)
                call zms(u2,u2,u2rms)
                call zms(u3,u3,u3rms)
                call zms(u1,u2,u12rms)
                call zms(u1,u3,u13rms)
                call zms(u2,u3,u23rms)
CCCCCCCCCCCCC  GO TO fOURIER SPACE CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC  TRANSFORM EACH OF THE FIELDS FROM U(Z,Y,X) TO U(Z,M,N)
CCCC     BUT FIRST LETS INITIALIZE THE FFTS.
                     call expo
                 call yxfour(u1)
                 call yxfour(u2)
                 call yxfour(u3)
CCCCCCCCCCCCC  COMPUTE CORRELATION  MATRICRES   CCCCCCCCCCCCCCCC
C  COMPUTE EACH OF THE SIX INDEPENDENT MEMBERS OF THE CORRELATION
C  MATRIX
           call matrix(u1,u1,u1u1,1)
           call matrix(u2,u2,u2u2,1)
           call matrix(u3,u3,u3u3,1)
           call matrix(u1,u2,u1u2,2)
           call matrix(u1,u3,u1u3,3)
           call matrix(u2,u3,u2u3,4)
 1              continue
CCCCCCCCCCCCCCCC   NOW SCALE RESULTS    CCCCCCCCCCCCCCCCCCCCCCCCCCCC
               con  = 1./(float(30)*fac)
               con1 = 1./(float(nens)*fac1)
           call scale(u1u1,con1)
           call scale(u2u2,con1)
           call scale(u3u3,con1)
           call scale(u1u2,con1)
           call scale(u1u3,con1)
           call scale(u2u3,con1)
           call mult(u1m,con)
           call mult(u2m,con)
           call mult(u3m,con)
           call mult(u1rms,con)
           call mult(u2rms,con)
           call mult(u3rms,con)
           call mult(u12rms,con)
           call mult(u13rms,con)
           call mult(u23rms,con)
CCCCCCCCCCCCCCCCC    COMPUTE RMS VALUES CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
             do 2 k = 1,nz1
               u1rms(k) = sqrt(u1rms(k) - u1m(k)**2)
               u2rms(k) = sqrt(u2rms(k) - u2m(k)**2)
               u3rms(k) = sqrt(u3rms(k) - u3m(k)**2)
               u12rms(k) = (u12rms(k)- u1m(k)*u2m(k))
               u13rms(k) = (u13rms(k)- u1m(k)*u3m(k))
               u23rms(k) = (u23rms(k)- u2m(k)*u3m(k))
 2               continue
                u12rms(1) = 0.0
                u12rms(nz1) = 0.0
                u13rms(1) = 0.0
```

```
                u13rms(nz1) = 0.0
                u23rms(1) = 0.0
                u23rms(nz1) = 0.0
              do 3 k =2,nz1-1
                 u12rms(k) = u12rms(k)/(u1rms(k)*u2rms(k))
                 u13rms(k) = u13rms(k)/(u1rms(k)*u3rms(k))
                 u23rms(k) = u23rms(k)/(u2rms(k)*u3rms(k))
 3             continue
CCCCCCCCCCCCC  COMPUTE RMS VALUES FROM SPECRAL RESULTS CCCCCCCCCCCCCCC
C     HERE WE INTEGRATE THE CORRELATION FUNCTIONS (AFTER HAVING SCALED
C     THEM PROPERLY) TO SEE IF WE PERFORMED THE CALCULATION PROPERELY.
C     THAT IS, DOES THE RMS WE GET FROM THE SPECTRA EQUAL THE TRUE RMS
C     FOR EACH FIELD?
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
           call intg(u1u1,s1)
           call intg(u2u2,s2)
           call intg(u3u3,s3)
           call intg(u1u2,s12)
        do 4 k =1,nzd21
           s1(k) = sqrt(2.0 * s1(k))
           s2(k) = sqrt(2.0 * s2(k))
 4         s3(k) = sqrt(2.0 * s3(k))
                s12(1) = 0.0
           do 5 k = 2,nzd21
              s12(k) = (2.* s12(k))/(s1(k)*s2(k))
 5          continue
CCCCCCCCCCCCCCCCCC    WRITE RESULTS TO DISC CCCCCCCCCCCCCCCCCCCCC
C     FIRST DUMP THE CORRELATION MATRICES
C     NOTE WE ARE NOW DUMPING FROM IZ =1,NZD21
C     AND IZP = 1,NZ1. THAT IS ONE HALF OF EACH MATRIX.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
              do 10 izp =1,nz1
        write(50)(((u1u1(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
        write(51)(((u2u2(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
        write(52)(((u3u3(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
        write(53)(((u1u2(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
        write(54)(((u1u3(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
        write(55)(((u2u3(j,i,iz,izp),j=1,23),i=1,14),iz=1,nzd21)
 10           continue
C     THEN DUMP THE MEAN VALUES AND THE RMS VALUES OF EACH FIELD
      write(60,100)(u1m(k),k=1,nz1)
      write(61,100)(u2m(k),k=1,nz1)
      write(62,100)(u3m(k),k=1,nz1)
      write(63,100)(u1rms(k),k=1,nz1)
      write(64,100)(u2rms(k),k=1,nz1)
      write(65,100)(u3rms(k),k=1,nz1)
      write(66,100)(u12rms(k),k=1,nz1)
      write(67,100)(u13rms(k),k=1,nz1)
      write(68,100)(u23rms(k),k=1,nz1)
CCC  THEN DUMP THE MEAN SQARE VALUES COMPUTED FROM THE SPECTRA.
C     NOTE THAT SINCE THE SYMMETRIES WERE USED TO IN COMPUTING
C     THESE SPECTRA THE DEGENERACIES MUST BE INCLUDED IN THE
C     COMPUTATION OF THE ENERGIES. THIS HAS NOT BEEN DONE HERE
C     SO THAT THE ENERGY VALUES WE GET HERE WILL NOT AGREE
C     WITH THE VALUES WE GET FROM THE RAW DATA. THESE DEGENERACIES
C     CAN BE ACCOUNTED FOR AND THE RESULTS WHEN WE DO INCLUDE THEM
C     PROPERLY DO APEAR TO BE CORRECT.
        write(70,100)(s1(k),k=1,nzd21)
        write(71,100)(s2(k),k=1,nzd21)
        write(72,100)(s3(k),k=1,nzd21)
```

```
      write(73,100)(s12(k),k=1,nzd21)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 100  format(2x,4e12.5)
      stop
      end
      subroutine matrix(a,b,c,is)
      parameter(nz=32,ny=64,nx=16)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
      parameter(nyd2=ny/2,nyd21=ny/2+1)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
      dimension a(nz1,ny,nxp2),b(nz1,ny,nxp2),c(nyd2,nxp2,nzd21,nz1)
      dimension r1(ny,nxp2),r2(ny,nxp2)
C      IN THIS SUBROUTINE WE COMPUTE THE CORRELATION (C) FROM THE
C      FIELDS A AND B. WE UTILIZE ALL THE AVAILABLE SYMMETRIES (4) FOR
C      THE CHANNEL FLOW PROBLEM. FOR EACH PAIR OF WALL NORMAL INDICES
C      IZ AND IZP WE COMPUTE THEIR COUNTERPARTS IZ1 AND IZ2. THE INDEX
C      IS DETERMINES THE SIGNS TO BE USED WHEN THE SYMMETRIES ARE APPLIED.
C      THE SUBROUTINES SYM1 AND SYM2 ARE USED TO APPLY THE REFLECTIONAL
C      SYMMETRIES.     PLEASE NOTE!!! IZP GOES FROM 1 TO NZ1.
            do 1 iz = 1,nzd21
                 iz1 = nz1 - iz + 1
            do 1  izp = 1,nz1
                 iz2 = nz1 - izp + 1
      do 2  j=1,ny
      do 2  i=1,nxd21
              ir = 2*i -1
              im = 2*i
               w1r  =  a(iz,j,ir)
               w1im =  a(iz,j,im)
               w2r  =  b(izp,j,ir)
               w2im =  b(izp,j,im)
               w1pr  = a(iz1,j,ir)
               w1pim = a(iz1,j,im)
               w2pr  = b(iz2,j,ir)
               w2pim = b(iz2,j,im)
      r1(j,ir) = w1r*w2r + w1im*w2im
      r1(j,im) = w1r*w2im - w1im*w2r
      r2(j,ir) = w1pr*w2pr + w1pim*w2pim
      r2(j,im) = w1pr*w2pim - w1pim*w2pr
 2                continue
ccccccccccccccccccc   INVOKE SYMMETRIES ccccccccccccccccccccccccccccc
            if ( is .eq. 1 ) then
                call sym1(r1,ny,nxp2)
                call sym1(r2,ny,nxp2)
            do 3 i=1,nxp2
            do 3 j=1,nyd2
 3                r1(j,i) = r1(j,i) + r2(j,i)
                  endif
            if ( is .eq. 2 ) then
                call sym1(r1,ny,nxp2)
                call sym1(r2,ny,nxp2)
            do 4 i=1,nxp2
            do 4 j=1,nyd2
 4                r1(j,i) = r1(j,i) - r2(j,i)
                  endif
            if ( is .eq. 3 ) then
                call sym2(r1,ny,nxp2)
                call sym2(r2,ny,nxp2)
            do 5 i=1,nxp2
            do 5 j=1,nyd2
```

```
 5                 r1(j,i) = r1(j,i) + r2(j,i)
                    endif
                if ( is .eq. 4 ) then
                    call sym2(r1,ny,nxp2)
                    call sym2(r2,ny,nxp2)
                do 6 i=1,nxp2
                do 6 j=1,nyd2
 6                  r1(j,i) = r1(j,i) - r2(j,i)
                    endif
CCCCCCCCCCC     CALCULATE ENSEMBLE AVERAGE CCCCCCCCCCCCCCCCCCCCC
                do 7 i = 1,nxp2
                do 7 j = 1,nyd2
      c(j,i,iz,izp) = c(j,i,iz,izp) + r1(j,i)
 7              continue
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 1              continue
      return
      end
      subroutine setup
      parameter(ny=64,nx=16)
      parameter(nxp2=nx+2,nxd21=nx/2+1)
      parameter(nyt2=2*ny,nyd21=ny/2+1,nyd22=ny/2+2)
      common/param/fac,fac1
cccccc  --------  some parameters   ---------
      fac = float(nx*ny)
      fac1 = ((fac**2)*2)
      return
      end
      subroutine hfft(a,is)
      parameter(ny=64,nx=16)
      parameter(nxp2=nx+2,nxd21=nx/2+1)
      parameter(nyt2=2*ny,nyd21=ny/2+1,nyd22=ny/2+2)
      parameter(nx3p4=3*nx+4,ny5=5*ny,nxny=(nx+2)*ny)
      dimension a(ny,nxp2)
      common/index/ indx(nxp2),indy(nyt2)
      common/exp/ exrc(nx3p4),excr(nx3p4),eycc(ny5)
      common/work/ vecx(nxp2),vecy(nyt2),vecy1(nyt2)
          if (is.eq. -1) go to 500
      do 1 j=1,ny
      do 2 i=1,nxp2
        indx(i)=(i-1)*ny+j
 2    continue
        call gather(nx,vecx,a,indx)
        call rcfft2(0,1,nx,vecx,exrc,vecx)
        call scatter(nxp2,a,indx,vecx)
 1    continue
      do 3 i=1,nxd21
      do 4 j=1,ny
         j1=2*(j-1)+1
      indy(j1)=j+2*(i-1)*ny
      indy(j1+1)=indy(j1)+ny
  4   continue
      call gather(nyt2,vecy,a,indy)
      call cfft2(0,1,ny,vecy,eycc,vecy)
      call scatter(nyt2,a,indy,vecy)
  3   continue
      return
500   continue
      do 5 i=1,nxd21
      do 6 j=1,ny
```

```
         j1=2*(j-1)+1
        indy(j1)=j+2*(i-1)*ny
        indy(j1+1)=indy(j1)+ny
 6      continue
        call gather(nyt2,vecy,a,indy)
        call cfft2(0,-1,ny,vecy,eycc,vecy)
        call scatter(nyt2,a,indy,vecy)
 5      continue
        do 7 j=1,ny
        do 8 i=1,nxp2
          indx(i)=(i-1)*ny+j
 8      continue
          call gather(nxp2,vecx,a,indx)
          call crfft2(0,-1,nx,vecx,excr,vecx)
          call scatter(nx,a,indx,vecx)
 7      continue
        do 10 i=1,nx
        do 10 j=1,ny
           a(j,i)=a(j,i)/(2.*float(nx*ny))
 10     continue
        return
        end
        subroutine sym1(a,ny,nx)
        dimension a(ny,nx)
         nyd2=ny/2
         nyd21 =ny/2+1
             do 1 i=1,nx
             do 1 j=1,nyd2
                  if ( j .eq. 1) then
                       a(j,i) = 2.0*a(j,i)
                       else
                       a(j,i) = (a(j,i) + a(ny-j+2,i))
                       endif
 1               continue
        return
        end
        subroutine sym2(a,ny,nx)
CCCCCC   NOTE: BECAUSE OF THE SYMMETRY IN THIS CASE THE ZERO SPANWISE
CCCCCC   MODES MUST BE ZERO
        dimension a(ny,nx)
         nyd2=ny/2
         nyd21 =ny/2+1
             do 1 i=1,nx
             do 1 j=1,nyd2
                  if ( j .eq. 1) then
                       a(j,i) = 0.0
                       else
                     a(j,i) = (a(j,i) - a(ny-j+2,i))
                       endif
 1               continue
        return
        end
        subroutine scale(a,con)
        parameter(nz=32,ny=64,nx=16)
        parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
        parameter(nyd2=ny/2,nyd21=ny/2+1)
        parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
        dimension a(nyd2,nxp2,nzd21,nz1)
         do 1 izp = 1,nz1
         do 1 iz = 1,nzd21
```

```
            a(1,1,iz,izp) = 0.0
       do 1 i=1,nxp2
       do 1 j=1,nyd2
1           a(j,i,iz,izp) = a(j,i,iz,izp)*con
       return
       end
      subroutine mult(a,con)
      parameter(nz=32,ny=64,nx=16)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
      parameter(nyd2=ny/2,nyd21=ny/2+1)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
      dimension a(nz1)
      do 1 iz = 1,nz1
1           a(iz) = a(iz)*con
       return
       end
       subroutine yxfour(a)
       parameter(nz=32,ny=64,nx=16)
       parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
       parameter(nyd2=ny/2,nyd21=ny/2+1)
       parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
       dimension a(nz1,ny,nxp2),work(ny,nxp2)
              do 1 k =1,nz1
                   do 2 j =1,ny
                   do 2 i =1,nx
  2                    work(j,i) = a(k,j,i)
               call hfft(work,1)
                   do 3 j =1,ny
                   do 3 i =1,nxp2
  3                    a(k,j,i) = work(j,i)
  1           continue
       return
       end
       subroutine mean(a,s)
       parameter(nz=32,ny=64,nx=16)
       parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
       parameter(nyd2=ny/2,nyd21=ny/2+1)
       parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
       dimension a(nz1,ny,nxp2),s(nz1)
            do 2 i =1,nx
            do 2 j =1,ny
            do 2 k =1,nz1
2               s(k) = a(k,j,i) + s(k)
      return
      end
      subroutine zms(a,b,s)
      parameter(nz=32,ny=64,nx=16)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
      parameter(nyd2=ny/2,nyd21=ny/2+1)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
      dimension a(nz1,ny,nxp2),b(nz1,ny,nxp2),s(nz1)
            do 2 i =1,nx
            do 2 j =1,ny
            do 2 k =1,nz1
2               s(k) = a(k,j,i)*b(k,j,i) + s(k)
      return
      end
      subroutine intg(a,s)
      parameter(nz=32,ny=64,nx=16)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
```

```
      parameter(nyd2=ny/2,nyd21=ny/2+1)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1)
      dimension a(nyd2,nxp2,nzd21,nz1),s(nzd21)
       do 1 iz = 1,nzd21
          izp = iz
       do 1 i=1,nxd21
       do 1 j=1,nyd2
          ir = 2*i-1
1            s(iz) = a(j,ir,iz,izp) + s(iz)
       return
       end
      subroutine expo
      parameter(ny=64,nx=16)
      parameter(nxp2=nx+2,nxd21=nx/2+1)
      parameter(nyt2=2*ny,nyd21=ny/2+1,nyd22=ny/2+2)
      parameter(nx3p4=3*nx+4,ny5=5*ny,nxny=(nx+2)*ny)
      common/work/vecx(nxp2),vecy(nyt2),vecy1(nyt2)
      common/exp/exrc(nx3p4),excr(nx3p4),eycc(ny5)
          call rcfft2(1,1,nx,vecx,exrc,vecx)
          call crfft2(1,1,nx,vecx,excr,vecx)
          call cfft2(1,1,ny,vecy,eycc,vecy)
       return
       end
```

```
      program klana
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   As described in the text, this code computes the
c   spatial correlation function by Fourier transformation of the
c   spectral correlations computed from the raw velocity fields
c   by using the klexp code.
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      parameter(nz=32,ny=64,nx=16)
      parameter(nym=23,nxm=14)
      parameter(nz1=nz+1,nzm1=nz-1,nzd21=nz/2+1,nzd2=nz/2)
      parameter(nyd2=ny/2,nyd21=ny/2+1,nymt2=nym*2)
      parameter(nxp2=nx+2,nxd2=nx/2,nxd21=nx/2+1,nxmd2=nxm/2)
      dimension z(nz1),cnorm(nzd21),cn(3,nz1)
      dimension u1u1(nym,nxm,nzd21,nz1),u2u2(nym,nxm,nzd21,nz1)
      dimension u3u3(nym,nxm,nzd21,nz1),u1u2(nym,nxm,nzd21,nz1)
      dimension u1u3(nym,nxm,nzd21,nz1),u2u3(nym,nxm,nzd21,nz1)
      dimension phi(nym,nxm),phisym(ny,nxp2),r(nyd2,nx)
      dimension rrij(nyd2,nx,nzd21,nz1)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    First read in the six components of the spectral correlation
c    function.
             do 10 izp = 1,nz1
      read(10)(((u1u1(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
      read(11)(((u2u2(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
      read(12)(((u3u3(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
      read(13)(((u1u2(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
      read(14)(((u1u3(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
      read(15)(((u2u3(j,i,iz,izp),j=1,nym),i=1,nxm),iz=1,nzd21)
  10          continue

      DO 8000 IJ=1,6   ! loop on 11,22,33,12,13,23 & calculate rrij

      DO 5010 IZP=1,NZ1
      do 5010 iz=1,nzd21

        GOTO (5021,5022,5023,5024,5025,5026), IJ

 5021   do 5031 j=1,nym
        do 5031 i=1,nxm
          phi(j,i)=u1u1(j,i,iz,izp)
 5031   CONTINUE
        goto 5029

 5022   do 5032 j=1,nym
        do 5032 i=1,nxm
          phi(j,i)=u2u2(j,i,iz,izp)
 5032   CONTINUE
        goto 5029

 5023   do 5033 j=1,nym
        do 5033 i=1,nxm
          phi(j,i)=u3u3(j,i,iz,izp)
 5033  CONTINUE
        goto 5029

 5024   do 5034 j=1,nym
        do 5034 i=1,nxm
          phi(j,i)=u1u2(j,i,iz,izp)
```

```
 5034   CONTINUE
        goto 5029

 5025   do 5035 j=1,nym
        do 5035 i=1,nxm
          phi(j,i)=u1u3(j,i,iz,izp)
 5035   CONTINUE
        goto 5029

 5026   do 5036 j=1,nym
        do 5036 i=1,nxm
          phi(j,i)=u2u3(j,i,iz,izp)
 5036   CONTINUE

 5029   continue

        call setphi(phi,phisym,IJ)
        call rij(phisym,r,cn,ij,iz,izp)
c
c NOW PUT r BACK INTO rrij - ("rij" is name of subroutine)
c
      do 5030 i=1,nx
      do 5030 j=1,nyd2
        rrij(j,i,iz,izp)=r(j,i)
 5030 continue
 5010 continue
 6000 CONTINUE
C
C Normalize each rrij, then write it to disk:
C
      GOTO (7501,7501,7501,7504,7505,7506), IJ

 7501 DO 7511 IZP=1,NZ1
      DO 7511 IZ=1,NZD21
      DO 7511 I=1,NX
      DO 7511 J=1,NYD2
        RRIJ(J,I,IZ,IZP)=RRIJ(J,I,IZ,IZP)/SQRT(CN(IJ,IZ)*CN(IJ,IZP))
 7511 CONTINUE
      GOTO 7600

 7504 DO 7514 IZP=1,NZ1
      DO 7514 IZ=1,NZD21
      DO 7514 I=1,NX
      DO 7514 J=1,NYD2
        RRIJ(J,I,IZ,IZP)=RRIJ(J,I,IZ,IZP)/SQRT(CN(1,IZ)*CN(2,IZP))
 7514 CONTINUE
      GOTO 7600

 7505 DO 7515 IZP=1,NZ1
      DO 7515 IZ=1,NZD21
      DO 7515 I=1,NX
      DO 7515 J=1,NYD2
        RRIJ(J,I,IZ,IZP)=RRIJ(J,I,IZ,IZP)/SQRT(CN(1,IZ)*CN(3,IZP))
 7515 CONTINUE
      GOTO 7600

 7506 DO 7516 IZP=1,NZ1
      DO 7516 IZ=1,NZD21
      DO 7516 I=1,NX
      DO 7516 J=1,NYD2
```

```
        RRIJ(J,I,IZ,IZP)=RRIJ(J,I,IZ,IZP)/SQRT(CN(2,IZ)*CN(3,IZP))
 7516 CONTINUE

 7600 CONTINUE
c
c DUMP rrij TO DISKFILE CONSISTENT WITH uij IN READ ROUTINE
c
      lu=80+ij-1
      write(lu)((((rrij(j,i,iz,izp),
     1                 j=1,nyd2),i=1,nx),iz=1,nzd21),izp=1,nz1)

8000  CONTINUE

      write(6,110)' NORMALIZATION FACTORS:'
  110 format(/a30)
      write(6,111) (((ij,izp,cn(ij,izp)),izp=1,nz1),IJ=1,3)
  111 format(5(2i6,f10.6))

      stop
      end

      subroutine setphi(a,b,IJ)

*************************************************************************
*  EXPANDS uiuj OR PHIij ARRAYS FROM a(nym,nxm) TO b(ny,nxp2)           *
*                          i. e., FROM a(23,14)   TO b(64,18)           *
*************************************************************************

      parameter (ny=64,nx=16)
      parameter (nym=23,nxm=14)
      parameter (nxp2=nx+2)
      dimension a(nym,nxm),b(ny,nxp2)

      do 1 i=1,nxm          !  1:14
      do 1 j=1,nym          !  1:23
    1   b(j,i)=a(j,i)

      do 2 i=nxm+1,nxp2     ! 15:18
      do 2 j=nym+1,33       ! 24:33
    2   b(j,i)=0.0

      do 3 i=nxm+1,nxp2     ! 15:18
      do 3 j=1,nym          !  1:23
    3   b(j,i)=0.0

      do 4 i=1,nxm          !  1,14
      do 4 j=nym+1,33       ! 24:33
    4   b(j,i)=0.0

      IF (IJ.LE.4) THEN     ! 11,22,33,12
      do 10 i=1,nxp2        !  1:18
      do 10 j=34,ny         ! 34:64
   10   b(j,i)= b(ny-j+2,i)
      ENDIF

      IF (IJ.GE.5) THEN     ! 13,23 have  odd symmetry!
      do 11 i=1,nxp2        !  1:18
      do 11 j=34,ny         ! 34:64
   11   b(j,i)=-b(ny-j+2,i)
      ENDIF
```

```
      return
      end


      subroutine rij(a,b,cn,ij,iz,izp)

*************************************************************************
*  INPUTS EXPANDED ARRAY FROM SETPHI, OUTPUTS CORRELATION FUNCTN. Rij *
*************************************************************************

      parameter (nz=32,nz1=nz+1,ny=64,nx=16)
      parameter (nxp2=nx+2,nxd21=nx/2+1)
      parameter (nyt2=2*ny,nyd21=ny/2+1)
      parameter (nyd2=ny/2,nxd2=nx/2,ny5=5*ny,nx3p4=3*nx+4)
      dimension eyy(ny5),exx(nx3p4)
      dimension vecy(nyt2),vecx(nxp2)
      dimension a(ny,nxp2),b(nyd2,nxp2),c(nyd2,nxp2),cn(3,nz1)

c
c  INITIALIZE FOURIER TRANSFORMS (CALC. eyy AND exx):
c
      call cfft2 (1,1,ny,vecy,eyy,vecy)
      call crfft2(1,1,nx,vecx,exx,vecx)
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN Y DIRECTION:
c
      do 1 i=1,nxd21

      do 2 j=1,ny
        vecy(2*j-1)=a(j,2*i-1)     ! REAL PART
    2   vecy(2*j  )=a(j,2*i  )     ! IMAG PART
        call cfft2(0,1,ny,vecy,eyy,vecy)

      do 3 j=1,nyd2                ! PUT VECY INTO TEMPORARY OUTPUT
        c(j,2*i-1)=vecy(2*j-1)
    3   c(j,2*i  )=vecy(2*j  )

    1 continue
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN X DIRECTION:
c
      do 4 j=1,nyd2

      do 5 i=1,nxp2
    5   vecx(i)=c(j,i)
      call crfft2(0,1,nx,vecx,exx,vecx)

      do 6 i=1,nx                  ! PUT VECX INTO FINAL OUTPUT
    6   c(j,i)=vecx(i)

    4 continue

      if (IJ.LE.3.AND.izp.eq.iz) then      ! set normalization factors
        cn(ij,iz)=c(1,1)                   ! ij=1,3 for 11,22,33
        if (cn(ij,iz).eq.0.) cn(ij,iz)=1.
        cn(ij,nz1+1-iz)=cn(ij,iz)
      endif

c
```

```
c  FINALLY, FLIP RIGHT SIDE TO THE LEFT OF THE LEFT SIDE:
c
      do 10 j=1,nyd2
      do 11 i=1,nxd2
        b(j,i  )=c(j,i+8)
   11   b(j,i+8)=c(j,i  )
   10 continue

      return
      end
```

```
      program wavspec

*********************************************************************
*     "Computes space-time corelations for planar data"             *
*   THIS VERSION READS 7 FILES, DOES 5 FFT'S, AND AVOIDS ZERO PACKING  *
*********************************************************************
*  This version reads three 3-d yxt files u1, u2, and u3, extracts from*
*  each three 2-d arrays (umn,unk,umk), does 2-d fft on each of those  *
*  still using same array names, and adds up the results over the      *
*  remaining  (third) variable.  It then calculates energies, ensemble *
*  averages, and computes correlation fundtions, saving all            *
*  results to disk.                                                     *
*********************************************************************

      parameter (ny=64,nx=16,ntim=512)
      dimension u1(ny,nx,ntim),u2(ny,nx,ntim),u3(ny,nx,ntim)
      dimension umn1(ny  ,  nx+2),umn2(ny  ,  nx+2),umn3(ny  ,  nx+2)
      dimension smn1(ny/2,nx/2+1),smn2(ny/2,nx/2+1),smn3(ny/2,nx/2+1)
      dimension smn1s(ny ,  nx+2),smn2s(ny ,  nx+2),smn3s(ny ,  nx+2)
      dimension rmn1(ny/2,  nx+2),rmn2(ny/2,  nx+2),rmn3(ny/2,  nx+2)
      dimension unk1(nx  ,ntim+2),unk2(nx  ,ntim+2),unk3(nx,  ntim+2)
      dimension snk1(nx,ntim/2+1),snk2(nx,ntim/2+1),snk3(nx,ntim/2+1)
      dimension snk1s(nx, ntim+2),snk2s(nx, ntim+2),snk3s(nx, ntim+2)
      dimension rnk1(nx/2,ntim+2),rnk2(nx/2,ntim+2),rnk3(nx/2,ntim+2)
      dimension umk1(ny,ntim+2)  ,umk2(ny,  ntim+2),umk3(ny,  ntim+2)
      dimension smk1(ny/2,ntim/2+1),smk2(ny/2,ntim/2+1),
     *          smk3(ny/2,ntim/2+1)
      dimension smk1s(ny,ntim+2) ,smk2s(ny,ntim+2) ,smk3s(ny,ntim+2)
      dimension rmk1(ny/2,ntim+2),rmk2(ny/2,ntim+2),rmk3(ny/2,ntim+2)
      dimension tmp(ny,nx),e(9),eavg(3),w(9),tmp1(nx,ntim/2+1)
c
c  initialize 9 ensemble averages smni, ski, smki, i=1,3 :
c
      do 40  m=1,ny/2
      do 40  n=1,nx/2+1
        smn1(m,n)=0.
        smn2(m,n)=0.
        smn3(m,n)=0.
   40 continue
      do 43  n=1,nx
      do 43  k=1,ntim/2+1
        snk1(n,k)=0.
        snk2(n,k)=0.
        snk3(n,k)=0.
   43 continue
      do 46 m=1,ny/2
      do 46 k=1,ntim/2+1
        smk1(m,k)=0.
        smk2(m,k)=0.
        smk3(m,k)=0.
   46 continue
c
c  7 files (lu=10,16) permit 5 FFT's with 240 k's left over at the end
c                                                                !====
      call read  (u1,u2,u3,tmp,nx,ny,ntim,10,  1,400)           ! 400
      call read  (u1,u2,u3,tmp,nx,ny,ntim,11,401,512)           ! 112
      call hfens (u1,u2,u3,umn1,umn2,umn3,smn1,smn2,smn3,unk1,unk2,unk3,
     * snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,nx,ny,ntim)  !====
      call read  (u1,u2,u3,tmp,nx,ny,ntim,11,  1,288)           ! 288
      call read  (u1,u2,u3,tmp,nx,ny,ntim,12,289,512)           ! 224
```

```
      call hfens (u1,u2,u3,umn1,umn2,umn3,smn1,smn2,smn3,unk1,unk2,unk3,
     * snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,nx,ny,ntim)    !====
      call read  (u1,u2,u3,tmp,nx,ny,ntim,12,  1,176)              ! 176
      call read  (u1,u2,u3,tmp,nx,ny,ntim,13,177,512)              ! 336
      call hfens (u1,u2,u3,umn1,umn2,umn3,smn1,smn2,smn3,unk1,unk2,unk3,
     * snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,nx,ny,ntim)    !====
      call read  (u1,u2,u3,tmp,nx,ny,ntim,13,  1, 64)              !  64
      call read  (u1,u2,u3,tmp,nx,ny,ntim,14, 65,464)              ! 400
      call read  (u1,u2,u3,tmp,nx,ny,ntim,15,465,512)              !  48
      call hfens (u1,u2,u3,umn1,umn2,umn3,smn1,smn2,smn3,unk1,unk2,unk3,
     * snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,nx,ny,ntim)    !====
      call read  (u1,u2,u3,tmp,nx,ny,ntim,15,  1,352)              ! 352
      call read  (u1,u2,u3,tmp,nx,ny,ntim,16,353,512)              ! 160
      call hfens (u1,u2,u3,umn1,umn2,umn3,smn1,smn2,smn3,unk1,unk2,unk3,
     * snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,nx,ny,ntim)    !====
c
c  Calculate and print energies:
c
      call energ1(smn1, ny, nx,   e(1),w(1))
      call energ1(smn2, ny, nx,   e(2),w(2))
      call energ1(smn3, ny, nx,   e(3),w(3))
      e(1)=sqrt(e(1)/5./512.)           !
      e(2)=sqrt(e(2)/5./512.)           !  average over 5*512 data pts
      e(3)=sqrt(e(3)/5./512.)           !
      print *, ' MN RMS values 1, 2, 3 are: ',(e(i),i=1,3)

      call energ2(snk1, nx, ntim, e(4),w(4))
      call energ2(snk2, nx, ntim, e(5),w(5))
      call energ2(snk3, nx, ntim, e(6),w(6))
      e(4)=sqrt(e(4)/5./64.*512./512.)  !
      e(5)=sqrt(e(5)/5./64.*512./512.)  !  energ2 divides by 512**2
      e(6)=sqrt(e(6)/5./64.*512./512.)  !
      print *, ' NK RMS values 4, 5./, 6 are: ',(e(i),i=4,6)

      call energ3(smk1, ny, ntim, e(7),w(7))
      call energ3(smk2, ny, ntim, e(8),w(8))
      call energ3(smk3, ny, ntim, e(9),w(9))
      e(7)=sqrt(e(7)/5./16.*512./512.)  !
      e(8)=sqrt(e(8)/5./16.*512./512.)  !  energ3 divides by 512**2
      e(9)=sqrt(e(9)/5./16.*512./512.)  !
      print *, ' MK RMS values 7, 8, 9 are: ',(e(i),i=7,9)
c
c  test energies:
c
      do 1050 i=1,3
        eavg(i)=0.
        do 1010 j=i,9,3
          eavg(i)=eavg(i)+e(j)
 1010   continue
        eavg(i)=eavg(i)/3.
        do 1020 j=i,9,3
          if(abs(e(j)-eavg(i))/eavg(i).gt.0.00001) then
          print *,' energy error! e(',j,') = ',e(j),';  avg = ',eavg(i)
          endif
 1020   continue
 1050 continue
c
c  normalize ensemble averages by energy:
c
      do 1100 m=1,ny/2
```

```
      do 1100 n=1,nx/2+1
        smn1(m,n)=smn1(m,n)/w(1)
        smn2(m,n)=smn2(m,n)/w(2)
        smn3(m,n)=smn3(m,n)/w(3)
 1100 continue
      do 1150 n=1,nx
      do 1150 k=1,ntim/2+1
        snk1(n,k)=snk1(n,k)/w(4)
        snk2(n,k)=snk2(n,k)/w(5)
        snk3(n,k)=snk3(n,k)/w(6)
 1150 continue
      do 1200 m=1,ny/2
      do 1200 k=1,ntim/2+1
        smk1(m,k)=smk1(m,k)/w(7)
        smk2(m,k)=smk2(m,k)/w(8)
        smk3(m,k)=smk3(m,k)/w(9)
 1200 continue
c
c  symmetrize smni, snki, smki:
c
      call setphi1(smn1,smn1s,ny,  nx)
      call setphi1(smn2,smn2s,ny,  nx)
      call setphi1(smn3,smn3s,ny,  nx)
      call setphi2(snk1,snk1s,nx,ntim)
      call setphi2(snk2,snk2s,nx,ntim)
      call setphi2(snk3,snk3s,nx,ntim)
      call setphi1(smk1,smk1s,ny,ntim)
      call setphi1(smk2,smk2s,ny,ntim)
      call setphi1(smk3,smk3s,ny,ntim)
c
c  calculate correlation functions:
c
      call rij1(smn1s,rmn1,ny,  nx)
      call rij1(smn2s,rmn2,ny,  nx)
      call rij1(smn3s,rmn3,ny,  nx)
      call rij2(snk1s,rnk1,nx,ntim)
      call rij2(snk2s,rnk2,nx,ntim)
      call rij2(snk3s,rnk3,nx,ntim)
      call rij3(smk1s,rmk1,ny,ntim)
      call rij3(smk2s,rmk2,ny,ntim)
      call rij3(smk3s,rmk3,ny,ntim)
c
c  flip snk spectra now that fft's are completed:
c
      do 1310 n=1,nx
      do 1310 k=1,ntim/2+1
        tmp1(n,k)=snk1(n,k)
 1310 continue
      do 1320 n=1,nx/2
      do 1320 k=1,ntim/2+1
        snk1(n,k)=tmp1(n+nx/2,k)
 1320 continue
      do 1330 n=nx/2+1,nx
      do 1330 k=1,ntim/2+1
        snk1(n,k)=tmp1(n-nx/2,k)
 1330 continue
      do 1410 n=1,nx
      do 1410 k=1,ntim/2+1
        tmp1(n,k)=snk2(n,k)
 1410 continue
```

```
      do 1420 n=1,nx/2
      do 1420 k=1,ntim/2+1
        snk2(n,k)=tmp1(n+nx/2,k)
 1420 continue
      do 1430 n=nx/2+1,nx
      do 1430 k=1,ntim/2+1
        snk2(n,k)=tmp1(n-nx/2,k)
 1430 continue
      do 1510 n=1,nx
      do 1510 k=1,ntim/2+1
        tmp1(n,k)=snk3(n,k)
 1510 continue
      do 1520 n=1,nx/2
      do 1520 k=1,ntim/2+1
        snk3(n,k)=tmp1(n+nx/2,k)
 1520 continue
      do 1530 n=nx/2+1,nx
      do 1530 k=1,ntim/2+1
        snk3(n,k)=tmp1(n-nx/2,k)
 1530 continue
c
c  save flipped (where applicable) spectra to disk:
c
      write(50,91) 1,e(1),eavg(1)
      write(50,92) ((smn1(m,n), m=1,ny/2), n=1,  nx/2+1)
      write(50,91) 2,e(2),eavg(2)
      write(50,92) ((smn2(m,n), m=1,ny/2), n=1,  nx/2+1)
      write(50,91) 3,e(3),eavg(3)
      write(50,92) ((smn3(m,n), m=1,ny/2), n=1,  nx/2+1)
      write(50,91) 4,e(4),eavg(1)
      write(50,92) ((snk1(n,k), n=1,  nx), k=1,ntim/2+1)
      write(50,91) 5,e(5),eavg(2)
      write(50,92) ((snk2(n,k), n=1,  nx), k=1,ntim/2+1)
      write(50,91) 6,e(6),eavg(3)
      write(50,92) ((snk3(n,k), n=1,  nx), k=1,ntim/2+1)
      write(50,91) 7,e(7),eavg(1)
      write(50,92) ((smk1(m,k), m=1,ny/2), k=1,ntim/2+1)
      write(50,91) 8,e(8),eavg(2)
      write(50,92) ((smk2(m,k), m=1,ny/2), k=1,ntim/2+1)
      write(50,91) 9,e(9),eavg(3)
      write(50,92) ((smk3(m,k), m=1,ny/2), k=1,ntim/2+1)

   91 format(2x,i3,2e12.5)
   92 format(2x,4e12.5)
c
c  save correlation functions:
c
      write(50,91) 1
      write(50,92) ((rmn1(m,n),m=1,ny/2),n=1,  nx)
      write(50,91) 2
      write(50,92) ((rmn2(m,n),m=1,ny/2),n=1,  nx)
      write(50,91) 3
      write(50,92) ((rmn3(m,n),m=1,ny/2),n=1,  nx)
      write(50,91) 4
      write(50,92) ((rnk1(n,k),n=1,nx/2),k=1,ntim)
      write(50,91) 5
      write(50,92) ((rnk2(n,k),n=1,nx/2),k=1,ntim)
      write(50,91) 6
      write(50,92) ((rnk3(n,k),n=1,nx/2),k=1,ntim)
      write(50,91) 7
```

```
      write(50,92) ((rmk1(m,k),m=1,ny/2),k=1,ntim)
      write(50,91) 8
      write(50,92) ((rmk2(m,k),m=1,ny/2),k=1,ntim)
      write(50,91) 9
      write(50,92) ((rmk3(m,k),m=1,ny/2),k=1,ntim)

      stop
      end



      subroutine read(u1,u2,u3,tmp,nx,ny,ntim,lu,ka,kz)
      dimension u1(ny,nx,ntim),u2(ny,nx,ntim),u3(ny,nx,ntim),tmp(ny,nx)
      do 50 k=ka,kz
        read (lu) ((u1(j,i,k),j=1,ny),i=1,nx)
        read (lu) ((u3(j,i,k),j=1,ny),i=1,nx)
        read (lu) ((u2(j,i,k),j=1,ny),i=1,nx)
        read (lu) ((tmp(j,i) ,j=1,ny),i=1,nx)
        read (lu) ((tmp(j,i) ,j=1,ny),i=1,nx)
        read (lu) ((tmp(j,i) ,j=1,ny),i=1,nx)
 50   continue
      return
      end



      subroutine hfens (u1,u2,u3,     umn1,umn2,umn3,smn1,smn2,smn3,
     *  unk1,unk2,unk3,snk1,snk2,snk3,umk1,umk2,umk3,smk1,smk2,smk3,
     *  nx,ny,ntim)

      dimension u1(ny,nx,ntim),u2(ny,nx,ntim),u3(ny,nx,ntim)
      dimension umn1(ny  ,  nx+2),umn2(ny  ,  nx+2),umn3(ny  ,  nx+2)
      dimension smn1(ny/2,nx/2+1),smn2(ny/2,nx/2+1),smn3(ny/2,nx/2+1)
      dimension unk1(nx  ,ntim+2),unk2(nx  ,ntim+2),unk3(nx,  ntim+2)
      dimension snk1(nx,ntim/2+1),snk2(nx,ntim/2+1),snk3(nx,ntim/2+1)
      dimension umk1(ny,ntim+2)  ,umk2(ny,  ntim+2),umk3(ny,  ntim+2)
      dimension smk1(ny/2,ntim/2+1),smk2(ny/2,ntim/2+1),
     *          smk3(ny/2,ntim/2+1)

c
c  At each time (3rd variable), extract umn array, fft it, compile sum:
c
        do 100 k=1,ntim
          do 70 m=1,ny
          do 70 n=1,nx
            umn1(m,n)=u1(m,n,k)
            umn2(m,n)=u2(m,n,k)
            umn3(m,n)=u3(m,n,k)
 70       continue
          call hfft1(umn1,ny,nx,1)
          call ensav1(umn1,smn1,ny,nx)
          call hfft1(umn2,ny,nx,1)
          call ensav1(umn2,smn2,ny,nx)
          call hfft1(umn3,ny,nx,1)
          call ensav1(umn3,smn3,ny,nx)
100   continue
c
c  At each y (3rd variable), extract unk array, fft it, compile sum:
c
        do 200 m=1,ny
```

```
            do 170 n=1,nx
            do 170 k=1,ntim
              unk1(n,k)=u1(m,n,k)
              unk2(n,k)=u2(m,n,k)
              unk3(n,k)=u3(m,n,k)
  170       continue
            call hfft2(unk1,nx,ntim,1)
            call ensav2(unk1,snk1,nx,ntim)
            call hfft2(unk2,nx,ntim,1)
            call ensav2(unk2,snk2,nx,ntim)
            call hfft2(unk3,nx,ntim,1)
            call ensav2(unk3,snk3,nx,ntim)
  200 continue
c
c  At each x (3rd variable), extract umk array, fft it, compile sum:
c
          do 300 n=1,nx
            do 270 m=1,ny
            do 270 k=1,ntim
              umk1(m,k)=u1(m,n,k)
              umk2(m,k)=u2(m,n,k)
              umk3(m,k)=u3(m,n,k)
  270       continue
            call hfft3(umk1,ny,ntim,1)
            call ensav3(umk1,smk1,ny,ntim)
            call hfft3(umk2,ny,ntim,1)
            call ensav3(umk2,smk2,ny,ntim)
            call hfft3(umk3,ny,ntim,1)
            call ensav3(umk3,smk3,ny,ntim)
  300 continue
      return
      end



      subroutine hfft1(a,n1,n2,is)
      parameter (m1=64,m2=16) ! needed to dimension noncalled arrays
      dimension a(n1,n2+2)
      dimension indx(m2+2),indy(2*m1)
      dimension exx(3*m2+4),exy(5*m1)
      dimension vecx(m2+2),vecy(2*m1),vecy1(2*m1)
          if (is.eq. -1) go to 500
      call rcfft2(1,1,n2,vecx,exx,vecx)
      call cfft2(1,1,n1,vecy,exy,vecy)
      do 1 j=1,n1
      do 2 i=1,n2+2
        indx(i)=(i-1)*n1+j
  2   continue
        call gather(n2,vecx,a,indx)
        call rcfft2(0,1,n2,vecx,exx,vecx)
        call scatter(n2+2,a,indx,vecx)
  1   continue
      do 3 i=1,n2/2+1
      do 4 j=1,n1
         j1=2*(j-1)+1
      indy(j1)=j+2*(i-1)*n1
      indy(j1+1)=indy(j1)+n1
  4   continue
      call gather(2*n1,vecy,a,indy)
      call cfft2(0,1,n1,vecy,exy,vecy)
```

```
      call scatter(2*n1,a,indy,vecy)
 3    continue
      return
500   continue
      call crfft2(1,-1,n2,vecx,exx,vecx)
      call cfft2(1,-1,n1,vecy,exy,vecy)
      do 5 i=1,n2/2+1
      do 6 j=1,n1
         j1=2*(j-1)+1
      indy(j1)=j+2*(i-1)*n1
      indy(j1+1)=indy(j1)+n1
6     continue
      call gather(2*n1,vecy,a,indy)
      call cfft2(0,-1,n1,vecy,exy,vecy)
      call scatter(2*n1,a,indy,vecy)
5     continue
      do 7 j=1,n1
      do 8 i=1,n2+2
        indx(i)=(i-1)*n1+j
8     continue
        call gather(n2+2,vecx,a,indx)
        call crfft2(0,-1,n2,vecx,exx,vecx)
        call scatter(n2,a,indx,vecx)
7     continue
      do 10 i=1,n2
      do 10 j=1,n1
         a(j,i)=a(j,i)/(2.*float(n2*n1))
10    continue
      return
      end


      subroutine hfft2(a,n1,n2,is)
      parameter (m1=16,m2=512) ! needed to dimension noncalled arrays
      dimension a(n1,n2+2)
      dimension indx(m2+2),indy(2*m1)
      dimension exx(3*m2+4),exy(5*m1)
      dimension vecx(m2+2),vecy(2*m1),vecy1(2*m1)
          if (is.eq. -1) go to 500
      call rcfft2(1,1,n2,vecx,exx,vecx)
      call cfft2(1,1,n1,vecy,exy,vecy)
      do 1 j=1,n1
      do 2 i=1,n2+2
        indx(i)=(i-1)*n1+j
2     continue
        call gather(n2,vecx,a,indx)
        call rcfft2(0,1,n2,vecx,exx,vecx)
        call scatter(n2+2,a,indx,vecx)
1     continue
      do 3 i=1,n2/2+1
      do 4 j=1,n1
         j1=2*(j-1)+1
      indy(j1)=j+2*(i-1)*n1
      indy(j1+1)=indy(j1)+n1
 4    continue
      call gather(2*n1,vecy,a,indy)
      call cfft2(0,1,n1,vecy,exy,vecy)
      call scatter(2*n1,a,indy,vecy)
 3    continue
      return
```

```
500  continue
     call crfft2(1,-1,n2,vecx,exx,vecx)
     call cfft2(1,-1,n1,vecy,exy,vecy)
     do 5 i=1,n2/2+1
     do 6 j=1,n1
        j1=2*(j-1)+1
     indy(j1)=j+2*(i-1)*n1
     indy(j1+1)=indy(j1)+n1
6    continue
     call gather(2*n1,vecy,a,indy)
     call cfft2(0,-1,n1,vecy,exy,vecy)
     call scatter(2*n1,a,indy,vecy)
5    continue
     do 7 j=1,n1
     do 8 i=1,n2+2
       indx(i)=(i-1)*n1+j
8    continue
       call gather(n2+2,vecx,a,indx)
       call crfft2(0,-1,n2,vecx,exx,vecx)
       call scatter(n2,a,indx,vecx)
7    continue
     do 10 i=1,n2
     do 10 j=1,n1
        a(j,i)=a(j,i)/(2.*float(n2*n1))
10   continue
     return
     end


     subroutine hfft3(a,n1,n2,is)
     parameter (m1=64,m2=512) ! needed to dimension noncalled arrays
     dimension a(n1,n2+2)
     dimension indx(m2+2),indy(2*m1)
     dimension exx(3*m2+4),exy(5*m1)
     dimension vecx(m2+2),vecy(2*m1),vecy1(2*m1)
         if (is.eq. -1) go to 500
     call rcfft2(1,1,n2,vecx,exx,vecx)
     call cfft2(1,1,n1,vecy,exy,vecy)
     do 1 j=1,n1
     do 2 i=1,n2+2
       indx(i)=(i-1)*n1+j
2    continue
       call gather(n2,vecx,a,indx)
       call rcfft2(0,1,n2,vecx,exx,vecx)
       call scatter(n2+2,a,indx,vecx)
1    continue
     do 3 i=1,n2/2+1
     do 4 j=1,n1
        j1=2*(j-1)+1
     indy(j1)=j+2*(i-1)*n1
     indy(j1+1)=indy(j1)+n1
 4   continue
     call gather(2*n1,vecy,a,indy)
     call cfft2(0,1,n1,vecy,exy,vecy)
     call scatter(2*n1,a,indy,vecy)
 3   continue
     return
500  continue
     call crfft2(1,-1,n2,vecx,exx,vecx)
     call cfft2(1,-1,n1,vecy,exy,vecy)
```

```
      do 5 i=1,n2/2+1
      do 6 j=1,n1
         j1=2*(j-1)+1
      indy(j1)=j+2*(i-1)*n1
      indy(j1+1)=indy(j1)+n1
6     continue
      call gather(2*n1,vecy,a,indy)
      call cfft2(0,-1,n1,vecy,exy,vecy)
      call scatter(2*n1,a,indy,vecy)
5     continue
      do 7 j=1,n1
      do 8 i=1,n2+2
        indx(i)=(i-1)*n1+j
8     continue
        call gather(n2+2,vecx,a,indx)
        call crfft2(0,-1,n2,vecx,exx,vecx)
        call scatter(n2,a,indx,vecx)
7     continue
      do 10 i=1,n2
      do 10 j=1,n1
         a(j,i)=a(j,i)/(2.*float(n2*n1))
10    continue
      return
      end


      subroutine ensav1(a,b,n1,n2)
      parameter (m1=64,m2=16)
      dimension a(n1,n2+2),b(n1/2,n2/2+1)
      dimension r(m1,m2/2+1) ! need var dim = r(n1,n2/2+1)
      do 1  i=1,n2/2+1
      do 1  j=1,n1
        r(j,i)=a(j,2*i-1)**2+a(j,2*i)**2
1     continue
      call sym1(r,n1,n2/2+1)
      do 2 i=1,n2/2+1
      do 2 j=1,n1/2
        b(j,i)=b(j,i)+r(j,i)
2     continue
      return
      end


      subroutine ensav2(a,b,n1,n2)
      dimension a(n1,n2+2),b(n1,n2/2+1)
      do 2  i=1,n2/2+1
      do 2  j=1,n1
        b(j,i)=b(j,i)+a(j,2*i-1)**2+a(j,2*i)**2
2     continue
      return
      end


      subroutine ensav3(a,b,n1,n2)
      parameter (m1=64,m2=512)
      dimension a(n1,n2+2),b(n1/2,n2/2+1)
      dimension r(m1,m2/2+1) ! need var dim = r(n1,n2/2+1)
      do 1 i=1,n2/2+1
      do 1 j=1,n1
        r(j,i)=a(j,2*i-1)**2+a(j,2*i)**2
```

```
1       continue
        call sym1(r,n1,n2/2+1)
        do 2 i=1,n2/2+1
        do 2 j=1,n1/2
          b(j,i)=b(j,i)+r(j,i)
2       continue
        return
        end


        subroutine sym1(c,m1,m2)
        dimension c(m1,m2)
        do 1 i=1,m2
        do 1 j=1,m1/2
          if ( j .eq. 1) then
            c(j,i) = c(j,i)
          else
            c(j,i) = (c(j,i) + c(m1-j+2,i))/2.
          endif
 1      continue
        return
        end


        subroutine energ1(b,n1,n2,e,w)
        parameter (m1=64,m2=16) ! needed to dimension noncalled arrays
        dimension b(n1/2,n2/2+1)
        dimension c(m1/2,m2/2+1)
        do 4 j=2,n1/2
          c(j,1)=b(j,1)/2.
          b(j,1)=b(j,1)/8.
      4 continue
        do 5 i=2,n2/2+1
          c(1,i)=b(1,i)/2.
          b(1,i)=b(1,i)/8.
      5 continue
        c(1,1)=0.
        b(1,1)=b(1,1)/16.
        do 6 i=2,n2/2+1
        do 6 j=2,n1/2
          c(j,i)=b(j,i)
          b(j,i)=b(j,i)/4.
      6 continue
        e=0.
        w=0.
        do 7 i=1,n2/2+1
        do 7 j=1,n1/2
          w=w+c(j,i)
          e=e+b(j,i)
      7 continue
        e=e*4./n1/n1/n2/n2
        return
        end


        subroutine energ2(b,n1,n2,e,w)
        parameter (m1=16,m2=512) ! needed to dimension noncalled arrays
        dimension b(n1,n2/2+1)
        dimension c(m1,m2/2+1)
        do 4 j=2,n1
```

```
      c(j,1)=b(j,1)/4.
      b(j,1)=b(j,1)/16.
    4 continue
      do 5 i=2,n2/2+1
      c(1,i)=b(1,i)/2.
      b(1,i)=b(1,i)/8.
    5 continue
      c(1,1)=0.
      b(1,1)=b(1,1)/16.
      do 6 i=2,n2/2+1
      do 6 j=2,n1
      c(j,i)=b(j,i)/2.
      b(j,i)=b(j,i)/8.
    6 continue
      w=0.
      e=0.
      do 7 i=1,n2/2+1
      do 7 j=1,n1
      w=w+c(j,i)
      e=e+b(j,i)
    7 continue
      e=e*4./n1/n1/n2/n2
      return
      end


      subroutine energ3(b,n1,n2,e,w)
      parameter (m1=64,m2=512) ! needed to dimension noncalled arrays
      dimension b(n1/2,n2/2+1)
      dimension c(m1/2,m2/2+1)
      do 4 j=2,n1/2
      c(j,1)=b(j,1)/2.
      b(j,1)=b(j,1)/8.
    4 continue
      do 5 i=2,n2/2+1
      c(1,i)=b(1,i)/2.
      b(1,i)=b(1,i)/8.
    5 continue
      c(1,1)=0.
      b(1,1)=b(1,1)/16.
      do 6 i=2,n2/2+1
      do 6 j=2,n1/2
      c(j,i)=b(j,i)
      b(j,i)=b(j,i)/4.
    6 continue
      e=0.
      w=0.
      do 7 i=1,n2/2+1
      do 7 j=1,n1/2
      w=w+c(j,i)
      e=e+b(j,i)
    7 continue
      e=e*4./n1/n1/n2/n2
      return
      end


      subroutine setphi1(a,b,n1,n2)
```

****

```
* EXPANDS smni   ARRAYS   FROM a(ny/2,nx/2+1)     TO b(ny,nx+2)          *
*                i. e.,   FROM a(32,       9)     TO b(64, 18)           *
*         smki   ARRAYS   FROM a(ny/2,ntim/2+1)  TO b(ny,ntim+2)        *
*                i. e.,   FROM a(32,     257)     TO b(64,514)           *
*****************************************************************************

      dimension a(n1/2,n2/2+1),b(n1,n2+2)

      do 1 i=1,n2/2+1       !  1:  9,257
      do 1 j=1,n1/2         !  1: 32, 32
        b(j,2*i-1) =a(j,i)  !  a into real(b)
        b(j,2*i  ) =0.      !  imag(b)=0.
    1 continue

      do 2 i=1,n2+2         !  1:  18,514
        b(n1/2+1,i)=0.      !  row 33, 33 = 0.
    2 continue

      do 10 i=1,n2+2        !  1: 10, 514
      do 10 j=n1/2+2,n1     ! 34:64, 34:64
        b(j,i)= b(n1-j+2,i)! symmetry for smni, smki
   10 continue
      b(1,1)=0.             ! elimininate (0,0) mode

      return
      end


      subroutine setphi2(a,b,n1,n2)

*****************************************************************************
* EXPANDS snki   ARRAYS   FROM a(nx,  ntim/2+1)  TO b(nx,  ntim+2)      *
*                i. e.,   FROM a(16,  257)       TO b(16,  514   )      *
*****************************************************************************

      dimension a(n1,n2/2+1),b(n1,n2+2)

      do 1 i=1,n2/2+1       !  1: 257
      do 1 j=1,n1           !  1: 16
        b(j,2*i-1) =a(j,i)  !  a into real(b)
        b(j,2*i  ) =0.      !  imag(b)=0.
    1 continue
      b(1,1)=0.             ! elimininate (0,0) mode

      return
      end


      subroutine rij1(a,b,n1,n2)

*****************************************************************************
*  INPUTS EXPANDED ARRAY FROM SETPHI, OUTPUTS CORRELATION FUNCTN. Rij *
*****************************************************************************

      parameter (m1=64,m2=16)    ! needed for uncalled array dimensions
      dimension eyy(5*m1),exx(3*m2+4)
      dimension vecy(2*m1),vecx(m2+2)
      dimension a(n1,n2+2),b(n1/2,n2+2),c(m1/2,m2+2)
c
c  INITIALIZE FOURIER TRANSFORMS (CALC. eyy AND exx):
```

```
c
      call cfft2 (1,1,n1,vecy,eyy,vecy)
      call crfft2(1,1,n2,vecx,exx,vecx)
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN Y DIRECTION:
c
      do 1 i=1,n2/2+1

      do 2 j=1,n1
        vecy(2*j-1)=a(j,2*i-1)     ! REAL PART
    2   vecy(2*j  )=a(j,2*i  )     ! IMAG PART
        call cfft2(0,1,n1,vecy,eyy,vecy)

      do 3 j=1,n1/2                ! PUT VECY INTO TEMPORARY OUTPUT
        c(j,2*i-1)=vecy(2*j-1)
    3   c(j,2*i  )=vecy(2*j  )

    1 continue
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN X DIRECTION:
c
      do 4 j=1,n1/2

      do 5 i=1,n2+2
    5   vecx(i)=c(j,i)
      call crfft2(0,1,n2,vecx,exx,vecx)

      do 6 i=1,n2                    ! PUT VECX INTO FINAL OUTPUT
    6   c(j,i)=vecx(i)

    4 continue

      cnorm=c(1,1)
      if (cnorm.eq.0.) cnorm=1.
c
c  FINALLY, FLIP RIGHT SIDE TO THE LEFT OF THE LEFT SIDE:
c
      do 10 j=1,n1/2
      do 11 i=1,n2/2
        b(j,i     )=c(j,i+n2/2)/cnorm
   11   b(j,i+n2/2)=c(j,i     )/cnorm
   10 continue

      return
      end


      subroutine rij2(a,b,n1,n2)

*************************************************************************
*  INPUTS EXPANDED ARRAY FROM SETPHI, OUTPUTS CORRELATION FUNCTN. Rij   *
*************************************************************************

      parameter (m1=16,m2=512)   ! needed for uncalled array dimensions
      dimension eyy(5*m1),exx(3*m2+4)
      dimension vecy(2*m1),vecx(m2+2)
      dimension a(n1,n2+2),b(n1/2,n2+2),c(m1/2,m2+2)
c
c  INITIALIZE FOURIER TRANSFORMS (CALC. eyy AND exx):
c
```

```
      call cfft2 (1,1,n1,vecy,eyy,vecy)
      call crfft2(1,1,n2,vecx,exx,vecx)
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN Y DIRECTION:
c
      do 1 i=1,n2/2+1

      do 2 j=1,n1
        vecy(2*j-1)=a(j,2*i-1)     ! REAL PART
    2   vecy(2*j  )=a(j,2*i  )     ! IMAG PART
        call cfft2(0,1,n1,vecy,eyy,vecy)

      do 3 j=1,n1/2                ! PUT VECY INTO TEMPORARY OUTPUT
        c(j,2*i-1)=vecy(2*j-1)
    3   c(j,2*i  )=vecy(2*j  )

    1 continue
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN X DIRECTION:
c
      do 4 j=1,n1/2

      do 5 i=1,n2+2
    5   vecx(i)=c(j,i)
      call crfft2(0,1,n2,vecx,exx,vecx)

      do 6 i=1,n2                  ! PUT VECX INTO FINAL OUTPUT
    6   c(j,i)=vecx(i)

    4 continue

      cnorm=c(1,1)
      if (cnorm.eq.0.) cnorm=1.
c
c  FINALLY, FLIP RIGHT SIDE TO THE LEFT OF THE LEFT SIDE:
c
      do 10 j=1,n1/2
      do 11 i=1,n2/2
        b(j,i     )=c(j,i+n2/2)/cnorm
   11   b(j,i+n2/2)=c(j,i     )/cnorm
   10 continue

      return
      end


      subroutine rij3(a,b,n1,n2)

*************************************************************************
*  INPUTS EXPANDED ARRAY FROM SETPHI, OUTPUTS CORRELATION FUNCTN. Rij *
*************************************************************************

      parameter (m1=64,m2=512)   ! needed for uncalled array dimensions
      dimension eyy(5*m1),exx(3*m2+4)
      dimension vecy(2*m1),vecx(m2+2)
      dimension a(n1,n2+2),b(n1/2,n2+2),c(m1/2,m2+2)
c
c  INITIALIZE FOURIER TRANSFORMS (CALC. eyy AND exx):
c
      call cfft2 (1,1,n1,vecy,eyy,vecy)
```

```
      call crfft2(1,1,n2,vecx,exx,vecx)
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN Y DIRECTION:
c
      do 1 i=1,n2/2+1

      do 2 j=1,n1
        vecy(2*j-1)=a(j,2*i-1)     ! REAL PART
    2   vecy(2*j  )=a(j,2*i  )     ! IMAG PART
        call cfft2(0,1,n1,vecy,eyy,vecy)

      do 3 j=1,n1/2                ! PUT VECY INTO TEMPORARY OUTPUT
        c(j,2*i-1)=vecy(2*j-1)
    3   c(j,2*i  )=vecy(2*j  )

    1 continue
c
c  EXTRACT VECTOR FROM ARRAY AND DO FFT IN X DIRECTION:
c
      do 4 j=1,n1/2

      do 5 i=1,n2+2
    5   vecx(i)=c(j,i)
        call crfft2(0,1,n2,vecx,exx,vecx)

      do 6 i=1,n2                    ! PUT VECX INTO FINAL OUTPUT
    6   c(j,i)=vecx(i)

    4 continue

      cnorm=c(1,1)
      if (cnorm.eq.0.) cnorm=1.

c
c  FINALLY, FLIP RIGHT SIDE TO THE LEFT OF THE LEFT SIDE:
c
      do 10 j=1,n1/2
      do 11 i=1,n2/2
        b(j,i     )=c(j,i+n2/2)/cnorm
   11   b(j,i+n2/2)=c(j,i     )/cnorm
   10 continue

      return
      end
```